Lab 2: Multiplexed 7-Segment Display

Introduction

In this lab you will learn how to use time multiplexing to efficiently use the I/O on your FPGA.

Learning Objectives

By the end of this lab you will have…

- Implemented a time-multiplexing scheme to drive two seven-segment displays with a single set of FPGA I/O pins.
- Built a simple transistor circuit to drive large currents from the FPGA pins.
- Practiced your ability to build Verilog systems in a modular way.

Requirements

Display two independent hexadecimal numbers on your dual seven-segment display. Use a DIP switch and four other input pins (possibly connected to a DIP switch on a breadboard) to provide the data for two hexadecimal numbers. You **must** use a single seven-segment decoder HDL module to drive the cathodes for both digits on the display, which therefore must be wired for multiplexed operation. Also, display the sum of the numbers on five LEDs. The seven segment display should be oriented to display the numbers upright to the viewer.

Lab 2 Specifications

Lab-specific Specifications

Proficiency

☐ HDL design includes only a single seven-segment decoder module.

- ☐ Sum of the numbers on the two displays is correctly displayed for all inputs
- ☐ Seven segment displays are same brightness regardless of how many segments are lit.

Excellence

- ☐ No noticeable bleeding of the digits between displays
- ☐ No flickering on the individual digits
- ☐ Current draw/sink on all FPGA pins are below the currents specified in the recommended operating conditions. Claims are backed up by calculations and reference to the appropriate items on the datasheet.
- ☐ Digits on the seven-segment display are upright to the viewer.

General Specifications

Proficiency

General Schematic Specifications

- ☐ All pin names labeled
- ☐ All pin numbers labeled
- ☐ Crossing wires clearly identified as junction or unconnected
- ☐ Neat layout (e.g., clear organization and spacing)
- ☐ All parts labeled with part number
- ☐ All component values present

Block Diagram

- ☐ Block diagram present with one block per SystemVerilog module
- ☐ Each block includes all input and output signals

HDL & Code Specifications

General Formatting

- ☐ Descriptive filename (e.g., `lab2_jb.sv`)
- ☐ Descriptive variable names
- ☐ Neat formatting (e.g., standard indentation, consistent formatting for variable names (kebab-case/snake_case/camelCase/PascalCase))
- ☐ Descriptive and clear function/module names

Comments

- ☐ Comments to indicate the purpose of each function/module

Lab Writeup/Summary

- ☐ Brief (e.g., 3-5 sentence) description of the main goals of the assignment and what was done.
- ☐ Explanation of design approach. How did you go about designing and implementing the design?
- ☐ Explanation of testing approach. How did you verify your design was behaving as expected?
- ☐ Statement of whether the design meets all the requirements. If not, list the shortcomings.
- ☐ Number of hours spent working on the lab are included.
- ☐ Writeup contains minimal spelling or grammar issues and any errors do not significantly detract from clarity of the writeup.
- ☐ (Optional) List commments or suggestions on what was particularly good about the assignment or what you think needs to change in future versions.

Excellence

General Schematic Specifications

- ☐ Standard symbols used for all components where applicable
- ☐ Signals "flow" from left to right where possible (e.g., inputs on left hand side, outputs on right hand side)
- ☐ Title block with author name, title, and date

HDL & Code Specifications

General Formatting

- ☐ Name, email, and date at the top of every file
- ☐ Comment at the top of each source code file to describe what is in it
- ☐ Clear and organized hierarchy (e.g., deliniation between top level modules and submodules)

Testbenches

- ☐ Testbenches written for each individual module to demonstrate proper operation
- ☐ Testbench output included in the report

Lab Writeup/Summary

- ☐ Writeup is free of spelling and grammar issues

Open specifications in new tab.

Discussion

Time-multiplexing is a technique to share a common expensive hardware resource for several purposes at different times. For example, the multicycle processor in E85 multiplexed the memory for both instruction and data access and multiplexed the ALU for data processing instructions, branch calculations, and program counter increments.

In this lab, you will time-multiplex your seven-segment decoder module to run both halves of a dual display. A convenient way to control which half is active is to turn ON the common anode of only one display at a time. The anode requires substantial current, more than an FPGA output pin can drive. You can use a transistor to drive the large current. The lab has a stock of 2N3906 PNP transistors suitable for this purpose. Be sure to limit the base current so that you don't draw too much current from the FPGA pin and choose a suitable switching speed. If you switch too slowly, your eye will notice the flicker. If you switch too fast for the electronics, the two digits will bleed together.

Hints

Look at your RTL schematic in your synthesis tool (`Tools -> Netlist Analyzer`). Understand why your code produces the hardware you see. Be sure your combinational logic doesn't have any registers. Be sure your logic has no latches or tristate buffers. The oscilloscope is handy for tracking down timing problems.